WHITEPAPER

Kata Containers Best Practices at Ant Group

OpenInfra Foundation

ANT GROUP

katacontainers

Overview

This white paper describes how Ant Group uses the new generation of infrastructure—Kubernetes and Kata Containers—to perform online/offline "co-location" (a technical method of hybrid deployment) and improve resource utilization. In this white paper, we will first briefly introduce the history and current situation of Ant Group's online/offline "co-location," including traditional solutions and problems. Afterwards, the white paper will describe how Ant Group used Kata Containers technology to solve these problems and realize an efficient, secure and stable online/offline "co-location." The white paper will also describe the resulting benefits, including helping Ant Group to achieve carbon neutrality by 2021.

Building a complex, robust and efficient distributed system requires complex system engineering, which cannot be completed by one or a few components. In this white paper, certain work on key components such as Kubernetes and the Linux kernel will also be discussed in relation to online/offline "co-location."

Background Introduction

Big Data Batch Computing Service

Ant Group is a typical data-driven company, where data plays a very important role in the entire process of business innovation, business change and decision-making. This requires a stable and powerful data processing platform to support the increasingly complex services.

The Ant Group Big Data Processing Platform supports TB/PB/EB distributed data processing, and provides developers and users with a complete data import solution and a variety of classic distributed computing models (such as MapReduce, Spark, etc.). The platform is used to solve a user's massive data computing problem and is applied to data analysis, mining, business intelligence and other application fields.

The Big Data Processing Platform primarily provides the storage and computing of batch structured data. It can provide solutions for massive data warehouses and analysis and modeling services for big data, especially for three types of big data processing scenarios:

- Building large-scale data warehouses and enterprise BI systems based on SQL;
- Developing big data applications based on distributed programming models such as DAG (similar to MapReduce) and Graph;
- Developing big data statistical models and data mining based on statistics and machine learning algorithms.

Co-location Technology

With the rapid growth of Ant Group's business in recent years, the company's required infrastructure resources are also increasing, which also brings corresponding increases in costs. The complex organizational structure, business model, and massive resources are accompanied by the increase in the complexity of the infrastructure and the contradiction between the resource utilization and the stable operation of the system.

In the past, in order to ensure the service quality of the business, Ant Group would provision enough resources for the online service to process user requests. However, this resource allocation method is very simple and exclusive, and the actual amount of resources allocated is generally far beyond the actual utilized resources by the application, which results in waste and a low utilization rate of resources. With the expansion of the user scale and the growth of the business volume, the cost pressure brought by this waste of resources is also increasing.

Online Service and Offline Service

Generally speaking, Ant Group has two types of services: online service and offline service.

The characteristics of online jobs include but are not limited to long running time, strong latency sensitivity, and tidal phenomenon that is influenced by people's common behavior and business operations activities. Typical applications include transactions and payments, advertising services, red envelope activities (cash or coupon promotions), and instant messaging.

Offline jobs have the characteristics of large computing requirements, high fault tolerance, insensitivity to delay, retryable, and long-term use of a large number of resources. A typical offline service is similar to jobs like Hadoop MapReduce, Spark, etc.

The following table summarizes the characteristics and differences between the two different types of service:

	Online Service	Online Service
Core Indicators	• Latency (Ms) • QPS • Request Error Rate	• Job Completion Rate • Job Completion Time
Resource Sensitivity	• Highest Latency Sensitivity. • High Resource Sensitivity And Insufficient Resources Will Lead To Slow Service Requests Or Even Errors, Which Will Have A Greater Impact.	• The Resource Sensitivity Is Relatively Low, The Fault Tolerance Rate Is High, And The Job Operation Error Can Be Retried, As Long As It Is Guaranteed To Be Completed At The Desired Time Point.
Traffic Peak	• Service Peaks In The Morning And Evening • Related To The User's Living Habits	• Average, Can Be Manually Controlled • Does Not Directly Affect Users
CPU Utilization	• High At Peak Time, Low At Idle Time	• Computing Maintains Sustained High CPU Utilization
Memory	• Low	• High
Disk I/O	• Low, Mostly Log I/O	• High, Large Amount Of Data Read And Write
Network	• High At Peak Time, Low At Idle Time, And Overall Lower Than Offline Services	• High

An offline computing service typically includes the following main features:

- Resource consumption: especially memory resources, which are much higher than ordinary online applications;
- Non-real-time: not sensitive to delay, allow probabilistic failure, and can retry;
- Can run off-peak with online computing: most of them can be run when the system is relatively idle, and can be peaks and valleys with the resource requirements of online services.

What's Co-location

Through the observation of the workload operating model, we found that the resource utilization of online service and offline computing service has a very obvious periodic pattern during the day and night. As shown below:

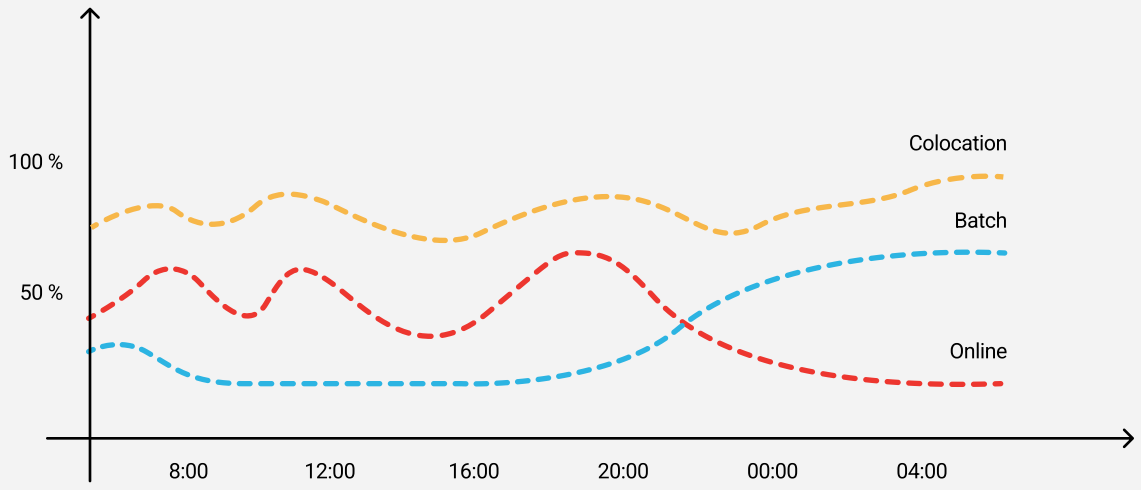

Figure 1: Online service and offline service resource usage

During the daytime, users are more active, online service requests are large, and different online services have different concentrated time periods. At this time, online services need to use more computing resources. At night, the number of active users decreases, the amount of online service requests decreases, the amount of computing resources required also decreases, and the resource utilization of online clusters also decreases.

Most of the batch computing service jobs are offline computing tasks, which require a large amount of resources. However, offline computing has a great tolerance for the use of resources. Even if the resources are not available for a short time, the jobs can continue to wait, as long as it can be completed within the limited time required by the user. This feature makes batch computing service very suitable for using resources that are idle at night in the cluster.

It can be seen from the above table and figure that different types of application resources have different usage characteristics. We can use these differences to hybridly deploy online services and offline jobs on the same physical resource. By using technical methods such as scheduling control and resource isolation, we eventually improve resource utilization and save costs on the basis of ensuring service stability. We call such a technology "co-location."

At the same time, different types of applications have different requirements for different types of resources, such as some requiring heavy CPU, some requiring heavy I/O or network. Through scheduling technology, services that rely on complementary resources are merged into the same node to achieve high-density deployment, which can also improve the utilization of system resources.

Through co-location, offline system and online system resources can also serve as elastic capacity for each other. When a big promotion event such as Double 11 requires a sudden increase in online service capacity, online services can occupy computing task resources to withstand the short-term ultra-high peak pressure, which, by this way, can not only flexibly solve the problem of insufficient computing resource for online services, but also reduce the overall cost of large promotions.

Core Indicators of the Co-location

There are three core indicators to measure the efficiency of a “co-location” system: isolation, efficiency, and cost.

Isolation

Isolation refers to the isolation of resources, faults, performance, and security between online and offline services with different characteristics, so that the two sides do not affect each other. In particular, applications that are primarily online have the highest priority. Offline applications should not affect the response time of online applications, nor interfere with the normal operation of online applications. Isolation means to reduce the delay and jitters to the online services caused by the offline services.

Efficiency

The operational efficiency of the offline service, is to ensure that the offline service operates efficiently and stably, and its operation performance meets the expected standard.

Cost

For infrastructure, cost is one of the most concerned core indicators. On the premise of ensuring the real-time (jitter and response time) of online applications and the performance of offline applications, the utilization and reuse rate of idle resources should be improved as much as possible.

Ant Group Co-location Status

With the introduction of cloud computing technology, more and more attention has been paid to the elasticity of IT infrastructure. In the past 10 years, the co-location technology of heterogeneous loads has been continuously tried. With the rapid development of container technology, online/offline co-location technology has also been able to be promoted more widely and deeply to a larger scale. Ant Group, together with the cooperating Alibaba team, achieved a large-scale online and offline co-location deployment in 2018. Up to now, Ant Group has achieved large-scale co-location of different types of services and computing. Technologies such as secure containers have brought a better balance between isolation effect and isolation overhead. The improvement of resource utilization and the stability of the system have made considerable contributions.

Problems of the Old Co-location System

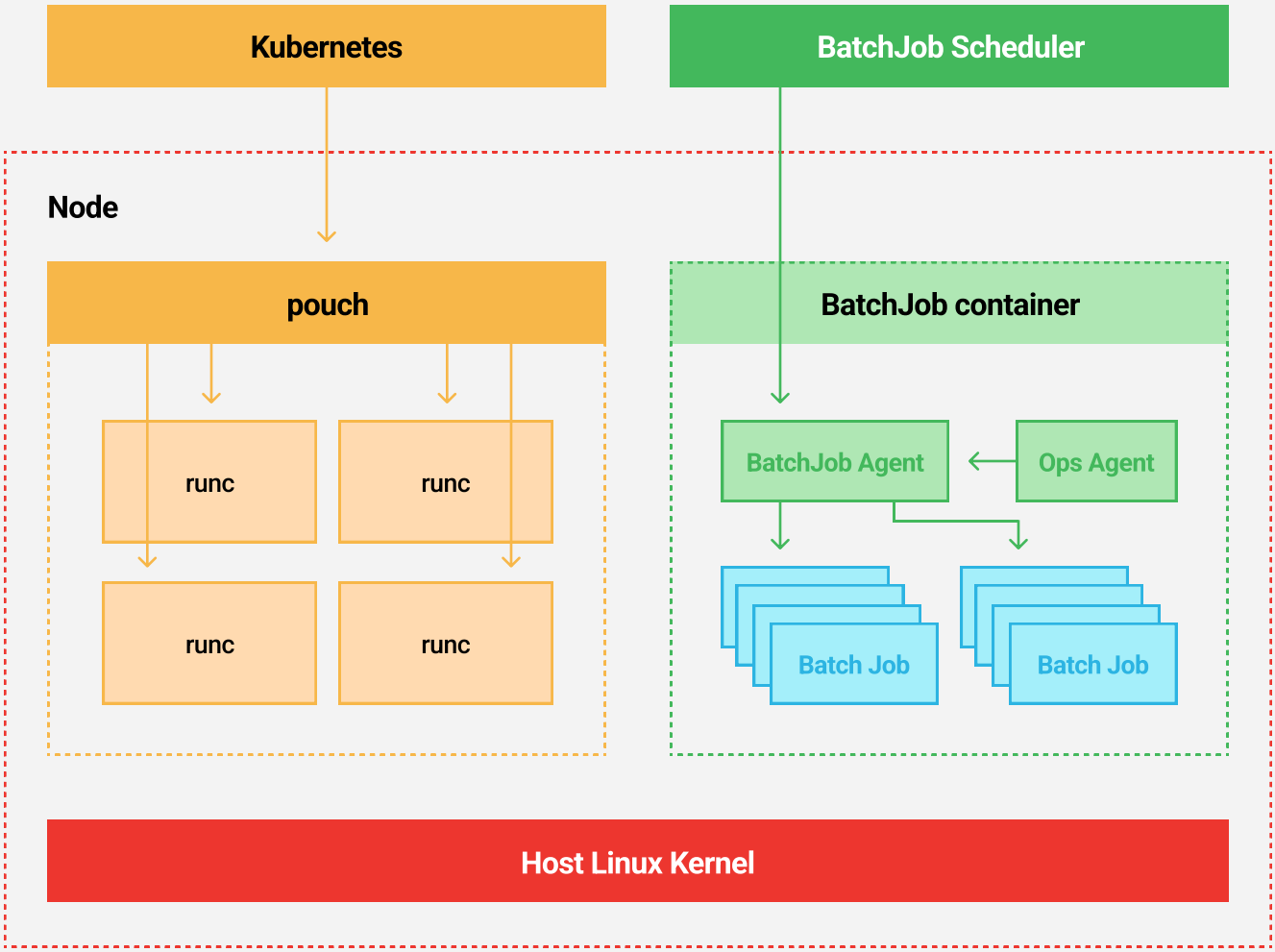

Figure 2: Old Co-location System

The diagram above shows the old Ant Group co-location framework based on Kubernetes and a batch scheduling system. This is a typical dual scheduler system. Batch computing jobs are jointly scheduled by two scheduling systems: Kubernetes and batch job scheduler. The two systems run on the same host and share the same Kernel.

Kubernetes is responsible for the life cycle management of online service Pods through Pouch. The containers in the Pod run under runc, and all runc containers share host resources.

The batch job scheduler schedules batch computing jobs on the host, and schedules offline jobs by allocating unused resources to online Pods in the host. Batch computing jobs and runc containers share the same host core and resources, increasing the number of tasks per unit node and improving resource utilization, but there is a competitive relationship between online and offline services in terms of resources.

Although the old co-location system has undergone long-term architecture optimization and stability management during the development process, there are still many problems that cannot be fundamentally solved under the current shared kernel architecture. When this kind of architecture also exposes some stability problems in actual operation, these problems also limit the further improvement of larger scale co-location and resource utilization.

The stability problems of the old batch computing service co-location system are mainly reflected in the following two aspects:

- Service stability
- Stability risks brought by operation and maintenance

Service Stability

Co-location pursues resource reuse, and sharing resources will inevitably lead to competition between online services and offline services, which will also affect the stability of both. Due to the characteristics of online services and the requirements of SLO (Service Level Objective), the problem of service stability mainly occurs on the online service side, especially the jitter of RT (Response Time) of online applications.

The core resource competition problems are mainly reflected in the following aspects:

CPU Resource Competition

In the old co-location environment, online services were deployed in the runc containers, and the cpuset mod was used for isolated deployment. Each online container monopolized a certain number of CPUs, and the containers did not interfere with each other. Offline batch computing service jobs are directly deployed on physical machines and share all CPU resources of the system with online containers. In order to avoid interference with online services, a series of isolation and optimization mechanisms have been adopted and restrictions and constraints applied to a large number of offline jobs running at various resource levels.

The kernel team has made a lot of improvements in scheduling and isolation; nevertheless, online and offline services still share the same host kernel, and offline services will create a large number of processes and threads, which will put huge pressure on CPU scheduling and bring interference to online services.

Memory Resource Competition

The competition of memory is reflected in the fact that the frequent memory release by offline services made a big impact on the kernel, resulting in a lot of stability problems. Although the impact can be reduced through functions such as limited memory allocation and release, the memory management risk still exists. Strong isolation of memory cannot be achieved under the premise of shared kernels.

I/O Resource Competition

The problem of I/O isolation has always been a difficulty in co-location stability, especially in shared disk clusters. Although we have been doing some optimizations based on shared kernels, we have not completely solved the problem of I/O resource competition.

Kernel Competition

There will be many problems with shared kernels. Competition between physical resources (memory pools, I/O resources, etc.) and kernel resources (locks, kernel paths, kernel thread services, etc.) is inevitable, resulting in isolation stability issues of memory, I/O, load, etc. For example, offline jobs occupy the kernel path for a long time, causing online services to be ready on the same HT (Hyper Thread) but unable to immediately preempt CPU resources.

Fault propagation

Because the kernels share the same swapper kernel thread, when a container OOM (Out-Of-Memory) occurs, it will affect all containers on the node.

Mutual performance interference

Due to the sharing of some kernel resources (such as kernel threads, etc.), different containers will interfere with each other even when the CPU and memory are isolated through cgroups.

To sum up, under the premise of a shared kernel, there is no perfect solution to the competition and isolation problems of various resources mentioned above.

Operations Stability

In the scheduling layer, since there are two different systems, there are also two independent ops systems belonging to different services. However, the two sets of ops systems directly face the same host resources, lack close connection with each other, and belong to a tightly coupled relationship. Therefore, when one SRE (Site Reliability Engineering) performs operations, it is likely to cause damage to the stability of the other services, such as hardware device management, software missing and incorrectly installed, software version conflicts, configuration conflicts, and kernel parameter conflicts. The operations problems caused by this may affect the normal operation of the services, or cause fatal system failures and affect the SLO of the entire business.

In the resource operations layer, the two separate ops systems use static resource division, which leads to resource fragmentation and reduces resource utilization.

Batch Computing Service Requirements for Co-location

From the above two points, the existing co-location technology cannot achieve complete performance and fault isolation in terms of architecture and cannot fundamentally solve the problem of stability risks caused by inconsistent service stability and operations. To solve these two problems, it is necessary to consider a completely different technical architecture at the root.

We need a better isolation solution to ensure business stability, and we need better decoupled host operations to improve ops efficiency and reduce ops risks, while supporting services that require different kernel versions to run on the same machine. Considering the scale of the cluster and facing a large number of nodes, we also need a more unified ops system.

Based on the characteristics of the batch computing service model and the coexistence of multiple schedulers, in order to build a new generation of co-location system that supports batch computing service, we have summarized the following requirements and prerequisites:

Multiple Priority

In the context of mutil-level scheduling co-location, the batch job scheduler itself also needs to schedule multi-priority batch computing service jobs. The batch job scheduler runs in a unified multi-priority security container on the host and schedules batch computing service jobs in the secure container, which requires the container to support multi-priority tasks.

Long-time Running

Also in the context of mutil-level scheduling co-location, a batch job scheduler scheduling framework needs to be started before any batch computing jobs. Although the batch computing jobs may have a short lifecycle, constantly being created and destroyed, the batch job scheduler, as a scheduling framework, needs to keep running for a long time like a daemon process on each host. Therefore, this requires that the batch computing service container can run for a long time without failure.

Reasons to choose Kata Containers

Based on the aforementioned shortcomings of the old co-location system, we believe that this problem needs to be solved fundamentally by architecture design. On one hand, a stronger isolation scheme is needed to reduce the impact of offline services on online services and ensure online services stability. On the other hand, it is necessary to decouple online service and offline service at the operation layer to reduce the risk introduced by different service operations, so as to avoid the impact on stability.

The emergence of Kata Containers, the secure containers, provides a new development direction for a new generation of co-location architectures. The strong isolation offered by Kata Containers can perfectly solve various interference problems caused by the shared kernel. Offering good compatibility with the container ecosystem, Kata Containers can be unified and managed by Kubernetes like traditional runc containers, seamlessly integrating into existing online services in the system architecture.

What is Kata Containers

Since the advent of Docker in 2013, container technology has immediately captivated developers around the world and has gradually become the mainstream way of building, publishing, and operating modern applications. Containers encapsulate applications in a standard format that can be quickly and securely switched from one computing environment to another, which is critical for developers who want to build, test, and deploy software quickly. However, the traditional container solution represented by runc is based on the shared kernel technology and is isolated and controlled through the cgroups and namespace solutions provided by Linux. The host system poses a serious threat. Especially in the public cloud environment, this potential threat has become a major obstacle to the popularization and implementation of container technology.

If different containers are nested into different virtual machines, by adding a layer of relatively safe and mature isolation technology, the security of the system can be greatly improved and the possibility of the system being breached can be reduced. Open source technologies based on this idea have been created: two representative projects are Clear Containers of Intel Open Source Technology Center and runV of Hyper.sh.

In 2017, the two open source projects merged to create Kata Containers, an open source project with the help of the then OpenStack Foundation (now Open Infrastructure Foundation, OIF), whose goal was to combine the security advantages of virtual machines and the high speed and manageability of containers, so as to provide users with standardized, secure and high-performance container solutions. Different Pods (containers) created by Kata Containers run in different virtual machines (kernels), providing better isolation and security than traditional containers, while inheriting the advantages of quick start and standardization of containers.

Kata Containers officially released version 1.0 in 2018, becoming the second top-level project of the Open Infrastructure Foundation after OpenStack. The Kata Containers project adheres to OpenInfra's open tradition of open code, open governance, and open design and is committed to promoting the development of basic container technologies with security features while collaborating across projects.

Features of Kata Containers

Kata Containers chose to use years-proven virtualization technology as the isolation layer, providing a more secure and isolated sandbox environment. At the same time, Kata Containers also provides a Docker-style user experience, which is sufficient in terms of ease of use. In addition, among the many features of Kata Containers are the following:

- Complete ecosystem compatibility. Kata Containers supports container runtime standards, including CRI (Container Runtime Interfaces) and OCI (Open Container Initiative), and integrates seamlessly with mainstream container orchestration platforms such as Kubernetes.
- The virtualization acceleration scheme of the underlying hardware is adopted and the performance is guaranteed.
- Supports multiple Hypervisors, such as QEMU, Firecracker, and Cloud-Hypervisor, providing users with multiple choices.
- Supports multiple CPU architectures, including x86, ARM, and s390.
- The product maturity is relatively high. Its original two security container projects are very mature. In particular, runV has begun to provide commercial services and has been verified by practice. Multiple enterprises are successfully using Kata Containers to achieve a secure and reliable container runtime.
- Open and active community environment. Active developers in the Kata Containers community are mainly from Ant Group, IBM (RedHat), ARM, Apple, Huawei and Intel, as well as Alibaba, Databricks, AMD, Baidu, Canonical, Google, NVIDIA, SUSE, Tencent, and Inspur. Companies such as China Mobile, China Unicom and ZTE have also made significant contributions.

Value of Kata Containers

Kata Containers provides an almost completely host-independent environment for accommodating the service within, including hardware and software system environments. In this way, it is compatible with the old offline job architecture to the greatest extent and helps offline jobs smoothly migrate to the safe container co-location solution. For example, the batch job scheduling framework of Ant Group relies on some system services to sense the resource situation of the system for adaptive job scheduling. Kata Containers can give them the required permissions and capabilities in the virtual machine container, so that they can be seamlessly migrated to run in Kata Containers without granting too many host permissions.

In addition, Kata Containers provides the following capabilities for the batch job scheduling framework:

Software environment compatibility

- Multi-priority CPU scheduling
- Independent and complete capability and syscall management
- Proc configuration
- Sysfs file system read and write
- Cgroup and namespace management
- Systemd services

Hardware environment compatibility

- CPU: Supports vPMU (virtual Performance Monitoring Unit), allowing real
- performance data to be collected through perf.
- Custom partition mount point
- Device passthrough
- Support job granularity CPI (Clocks Per Instruction) information collection
- Need to perceive the overall pressure of the physical server CPU and the
- overall pressure of the online services

Kata Containers, the secure containers, integrates the strong isolation of virtualization and the lightweight characteristics of containers, which can seamlessly connect with the container cloud infrastructure ecology, maintain the user experience of containers, and realize the complete cloud native-ization of the infrastructure.

Overall Architecture of the Kata Containers Solution

In 2019, the Ant Group secure container team, resource scheduling team, Alibaba Cloud OS Team, and Alibaba Group offline computing and search team jointly completed a Kata Containers unified co-location solution that supports security containers and multi-priority task scheduling. Comprising multiple components such as the underlying operating system, container runtime, container orchestration system, and batch computing service offline job scheduling system, the solution has opened up the batch computing service node management and control plane based on Kata Containers, including network, storage, and device plug-ins implemented by the container platform.

Overall Architecture

The co-location solution on the cloud is based on Kata Containers. It adopts the kernel isolation architecture. The online service containers continue to run under runc, while the offline applications run in Kata Containers with an independent kernel. This guest kernel only provides kernel requests for batch computing service jobs in Kata Containers and isolates batch computing service jobs from online service containers in terms of kernel resources and system calls, so that at the application level, online and offline services will not share the same kernel, which achieves stronger isolation. Batch computing service runs in Kata Containers, which can greatly reduce the complexity of operations and improve the stability of the co-location system under the premise of realizing the co-location reuse of host resources.

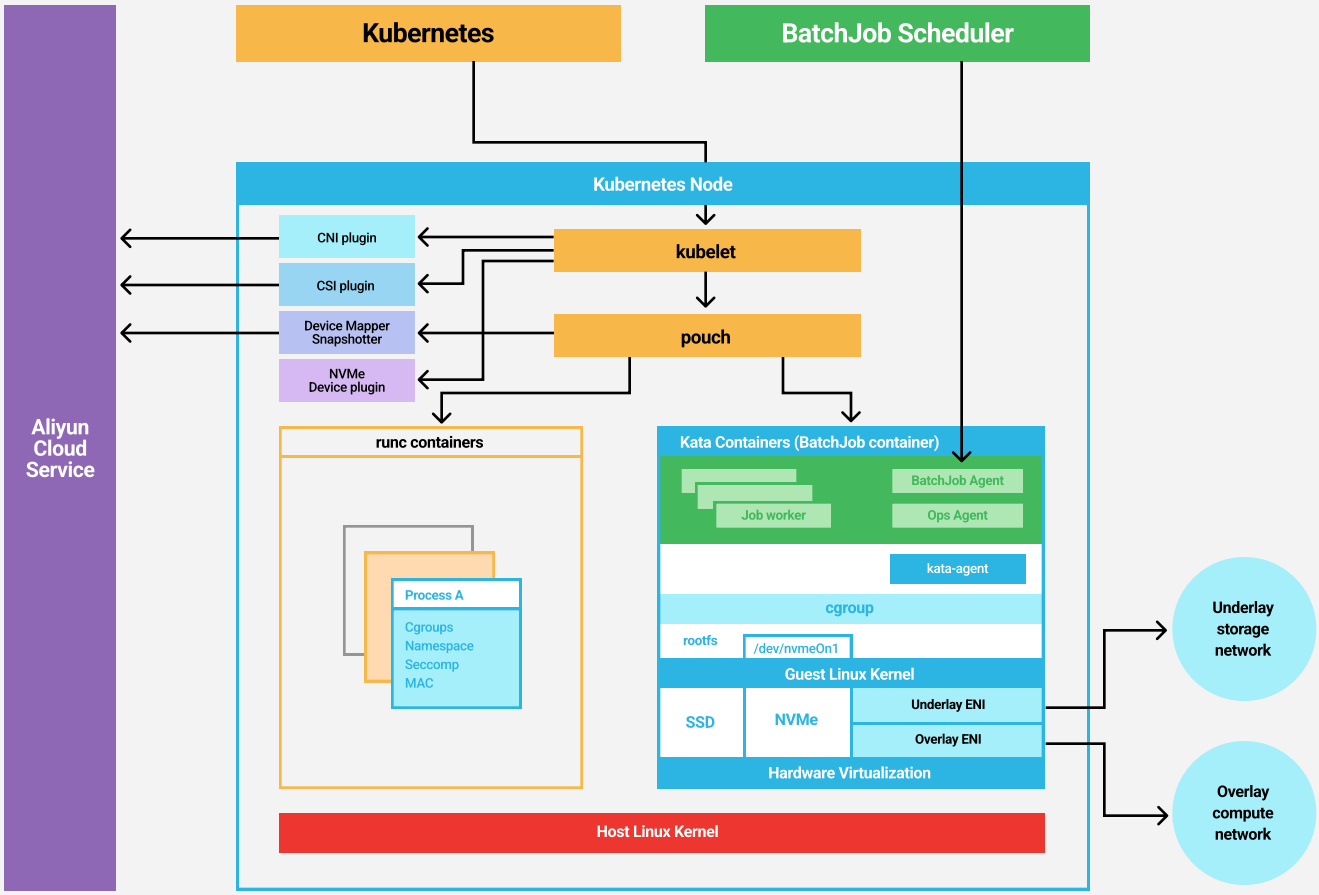

Figure 3: Ant Group Co-location Architecture Diagram

In the new co-location solution, the batch computing container runs the components required by the offline batch job scheduling framework. For the batch job scheduling framework, Kata Containers provides a complete virtual host for the batch computing container. It includes systemd, Ops agent and Batch job agent and also provides Linux core capabilities such as cgroup and namespace.

Kubernetes is responsible for the unified management and scheduling of the life cycle of batch computing containers, and the Ops agent is responsible for the deployment and monitoring of Batch job agents and storage systems in batch computing containers.

Key Features of Kata Containers For Co-location

Based on the particularity of Ant Group's specific business, we have enhanced and optimized Kata Containers in many aspects to support large-scale and stable online and offline services co-location.

Isolation Enhancement

Although Kata Containers has adopted a kernel-separated architecture—which isolates the system calls of the offline services to the operating system and allows the Kata Containers to use an exclusive network device and storage through the pass-through technology—the other two resources, memory and CPU, cannot be completely exclusive, but rather are shared with online services. In terms of these two resources, Kata Containers has done a lot of isolation enhancement and optimization, such as running the hypervisor process in offline priority and optimizing the EPT (Extended Page Table) to solve the high system workload problem caused by multi-core concurrent EPT page faults.

Multi-priority Jobs Support

There are two types of priorities: task priority and resource priority.
In the scheduling system of the operating system, the task priority reflects the importance of the task. Tasks with different priorities will have different processing manner, especially in the following two aspects:

- Resources allocation – high-priority tasks are easier to obtain resources for than low-priority tasks;
- Resource reclaiming – when resource contention occurs, the scheduler will first reclaim the resources of low-priority tasks and assign them to high-priority tasks.

In the dual scheduler scenario of batch computing service online and offline co-location, it is necessary to run batch computing service offline jobs with different priorities in the Kata Containers at the same time, which requires Kata Containers to support a multi-level priority mechanism in the guest environment.

Resource priority reflects the kernel scheduling QoS (Quality of Service) level, which refers to the degree to which the kernel guarantees resources in various aspects during runtime after the task passes the scheduling system. Resource priorities include the CPU's kernel scheduling priority and preemption priority, cached memory reclaim order, handling priority for OOM, and network DSCP (Differentiated Services Code Point).

Performance and Resource Optimization

Kata Containers uses the independent kernel architecture to isolate the access of offline jobs to host resources and reduces the interference of offline services to online services. But this completely isolated kernel architecture also brings some problems:

The cost doubles in terms of system calls, resource management, and allocation of the kernel in the guest environment, including host-guest two-layer scheduling and page table translation, storage and network virtualization, etc. Compared with runc that uses the shared kernel, Kata Containers will obviously have additional costs and performance overhead.

On the other hand, the independent kernel also isolates the memory of the host and Kata Containers guest from each other, which is not conducive to improve the overall resource utilization rate. From the perspective of cost, we actually just want to isolate the execution path of the kernel and the resources inside it, and for computing resources that do not belong to the kernel, we should try to maximize the sharing.

In order to solve the performance loss caused by the independent guest kernel, Kata Containers improves its performance and the overall resources utilization rate by using technologies such as rootfs and NVMe data disk pass-through based on network disk, network card pass-through, and ballon memory elasticity.

Pods Control Plane

Core technology of co-location is mainly divided into two aspects: one is isolation technology, and the other is resource scheduling technology. It can be said that the entire co-location structure of Ant Group is based on these two key points.

Control plane is mainly related to scheduling. From the time when the user creates a Pod until the Pod starts to run, this lifecycle needs to work collaboratively with the entire control plane.

Control Plane Architecture

The general components and structure of the entire control plane are as follows:

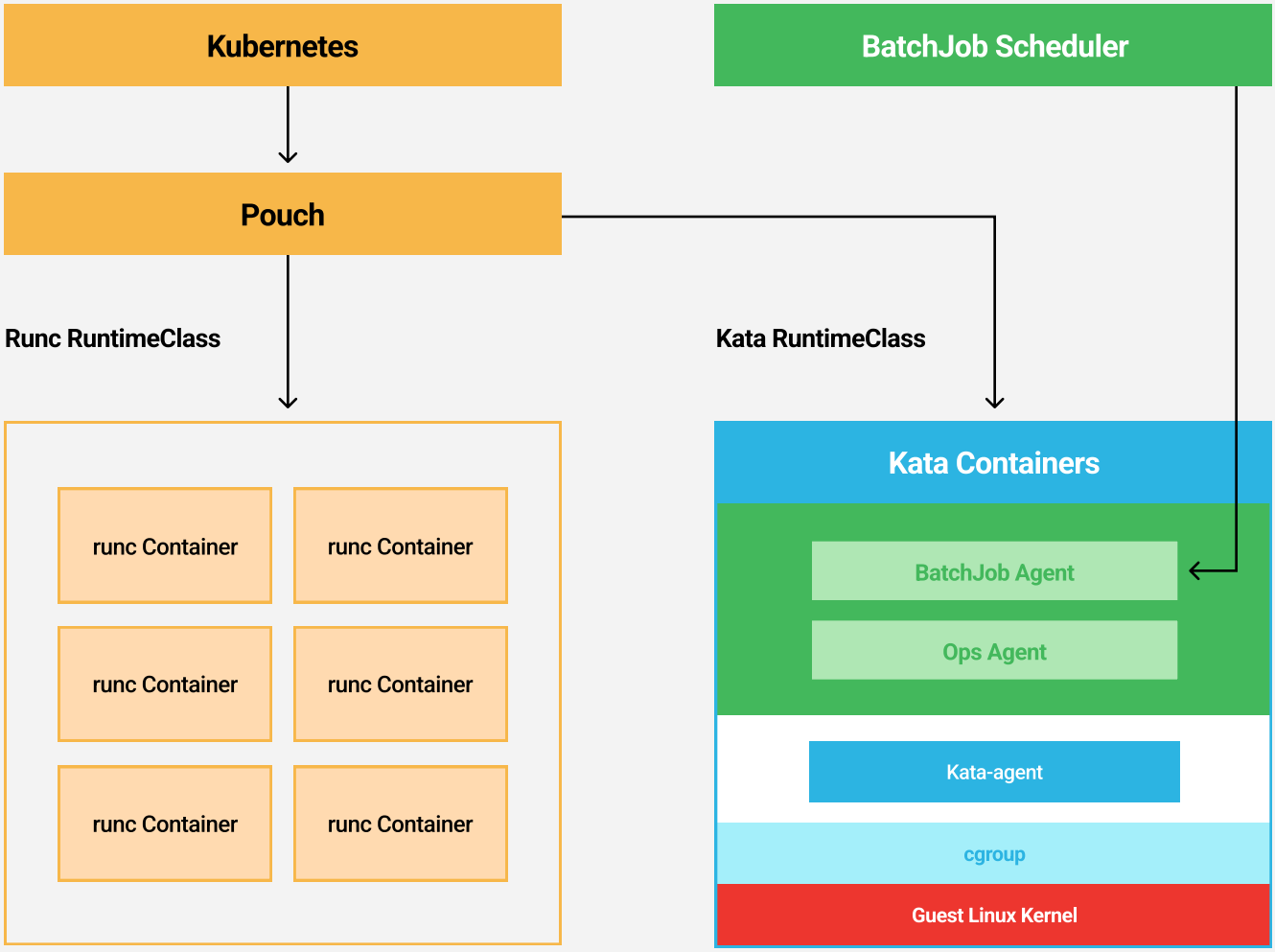

Figure 4: Kubernetes Control Plane

The entire system runs on a Kubernetes compatible cluster, and Kubernetes is responsible for the scheduling of Pods on the cluster. Pouch runs on each node and acts as the CRI runtime on the node and is responsible for managing the underlying containers.

In terms of the control plane, batch computing jobs are mainly scheduled by Kubernetes and the batch scheduling system in two layers. Kubernetes provides the Pod infrastructure for batch computing jobs and manages the lifecycle of the Pod. The batch computing scheduling system treats a Pod its own computing node, performs offline jobs scheduling in the Pod, and distributes jobs to the batch scheduling agent of the Pod (Batch job agent). The batch scheduling agent is responsible for creating offline computing jobs.

Kubernetes

In terms of resource scheduling, Kubernetes has grown into the mainstream ecological standard in the industry. However, although Kubernetes is good at online scheduling, it still has shortcomings in big data and ultra-large-scale scheduling, and it also lacks the best practices in the industry. Ant Group's Kubernetes team and Alibaba Group have added an online and offline unified scheduling platform that supports various business scenarios and integrates on the basis of keeping Kubernetes open and flexible in design, combined with its own special needs and the best practice experience of existing infrastructure. It has the co-location scheduling capability to create differentiated SLOs.

As the brain of Kubernetes, the scheduler provides the following values in large-scale container scheduling scenarios:

- High efficiency: supports a variety of computing and storage scenarios;
- Low cost: delivers cost-optimized resources;
- Stability: ensures the stability of service.

Runtime Class

Different container runtimes are used for different usage scenarios, especially in co-location services. A node may run different types of services, which requires a Kubernetes node to support different types of container runtimes. Initially Kubernetes only supported running the Docker runtime, which is also the default container runtime. With the emergence of more and more container runtimes such as Kata Containers, the Kubernetes community proposed CRI (Container Runtime Interface) as a protocol for communication between Kubernetes and the underlying container runtime as an application carrier. This seamlessly supports various compatible underlying runtime implementations and realizes the decoupling of runtime implementations and Kubernetes.

The container runtime class is the solution to this problem. The RuntimeClass resource defines and configures the runtime on the Kubernetes cluster and can support running the user's Pod using different container runtimes in the same cluster according to different choices.

Pod Overhead

In the virtualized container runtime represented by Kata Containers, the Pod runs in the virtual machine, so the Pod itself (that is, the virtual machine) may also occupy certain resources. These resources are additional resources required by the Pod control process, independent from user requested resources. Pod overhead is a feature used to define the resources required by the Pod infrastructure to distinguish it from user resources.

In Kubernetes, Pod overhead is set by the runtime class associated with the Pod's RuntimeClass. If the Pod Overhead feature is enabled, the scheduler will not only consider the total resources requested by the container itself, but also the additional overhead of the Pod itself when scheduling Pods. Before the introduction of the Pod overhead function, users could only add additional resources required by Pods to user containers in a custom way to ensure that Pods and business containers can obtain enough resources to run.

When the Kubernetes scheduler schedules a Pod, it considers the overhead of the Pod and the total amount of resources requested by all containers in the Pod. The scheduler adds up the resource requests of all containers and Pod overhead and then looks for nodes with sufficient CPU and memory resources to schedule.

Pouch

Pouch is a lightweight container technology open sourced by Alibaba Group in 2017. As an enhancement of containerd, the main functions of Pouch are as follows:

- Provides integration of multiple services, hooks and multiple container functions, compatible with traditional operations model;
- Enables P2P image distribution through Dragonfly, speeds up large-scale image distribution, and reduces network workload and the pressure on image center(Currently, our image-related extensions have been sent back to containerd upstream, and containerd already includes the nydus-snapshotter sub-project, which not only supports Dragonfly images download, but also offers more support for Lazyload image acceleration and richer image capabilities.);
- In terms of kernel compatibility, it is more in line with the status quo of enterprises. Considering the slow upgrade and low version of the kernel of most enterprise IT systems, it is best suited to the existing underlying infrastructure.

Isolation Technology

Since the emergence of co-location technology in large-scale clusters, in order to further improve resource utilization, the research on the sharing and isolation of various system kernel resources has never stopped. Operating system kernel and Kata Containers provide different isolation technologies for various resources in order to share resources, reduce resource competition, and meet various QoS guarantee requirements.

The main resources involved in isolation are: CPU, memory, I/O, network, and processor (LLC cache and memory bandwidth).

All resources can be shared and preempted. We use isolation technology to keep online/offline services free from interference to ensure service stability. We also must consider how to maximize utilization of each resource through sharing and preemption. Basically we need to meet the following requirements:

- When high-priority tasks require resources, they can quickly preempt and obtain resources.
- When there are idle resources, all tasks can obtain running resources and share them.
- Resources between different tasks are exclusive and unaffected by other tasks.

CPU Isolation

Standard CPU Isolation Technology

CPU Set

The old co-location system adopts the CPU set mode to allocate exclusive CPU cores to online applications and isolate offline applications from the CPU cores. Although this method has the strongest isolation, it also has a big disadvantage. Once the CPU is allocated to an online application, other applications can no longer use this core, which leads to very low resource utilization:

- Unable to achieve CPU overcommit.
- It is inconvenient to make in-place specification adjustments to achieve resource flexibility.
- The minimum scheduling resource granularity is one core, which is not fine enough.

CPU Quota

CPU quota is an isolation strategy for shared CPU resources of the kernel scheduler, using the cfs_quota_us and cfs_period_us parameters to allocate CPU time to the requested CPU quota.

CPU quota uses strict CPU resource control. After the job uses up the specified quota time, even if the CPU has a lot of idle resources, the job will suspend running instead of continuing to execute on the idle CPU, and the total execution time will not exceed the time set by quota. At the same time, if the quota is not used up in one cycle, the unused CPU time will not be accumulated to the next cycle. If the quota is exhausted in a cycle, the job will enter the throttle state. For latency sensitive jobs, it will inevitably affect the delay and cause jitter.

CPU Share

Unlike CPU quota, the CPU share strategy uses relative proportional control. When multiple tasks share the same CPU resources, the relative proportion of the occupied CPU resources can be controlled by the share parameter. CPU share is a dynamic control strategy. When some cgroups are idle, the remaining running tasks can still use the available CPU resources in proportion to maximize CPU resource utilization.

CPU share is a relative ratio; tasks with the same share value can evenly allocate CPU resources. If one task is in a non-running state, another task can use all CPU resources.

But the CPU share mode is not perfect. If all applications run on all CPU cores, it will bring significant performance degradation:

- Running multiple processes on one CPU core will inevitably cause context switching. Frequent process context switching means that the system consumes a lot of CPU.
- When processes switch back and forth in a large CPU sharepool, the cache hit rate of each CPU will be affected, L1/L2 Cache may be invalid, and each cache miss will cause performance loss.
- Kernel CFS scheduling also causes performance loss to a certain extent.

Isolation Scheme On CPU Resource Priority

Resource priority reflects the QoS level when the kernel schedules and allocates resources. It is used to provide different levels of control for different types of services, ensuring that high-priority tasks are easier to obtain resources for than low-priority tasks and can preempt resources of low-priority tasks when necessary, so as to ensure stable service.

The impact of resource priority is reflected in the degree of guarantee of various resource dimensions of the kernel. In terms of CPU resources, it mainly includes the priority of kernel scheduling and the priority of CPU preemption. Various CPU isolation mechanisms have different characteristics. At present, Kubernetes already supports all types of CPU resource isolation and allocation strategies, and the focus will be on the support for the isolation mechanism based on CPU share.

Eventually for CPU isolation of online and offline service Pods, we will adopt a tree structure of CPU set + CPU shares combination, whose design is shown in the following figure:

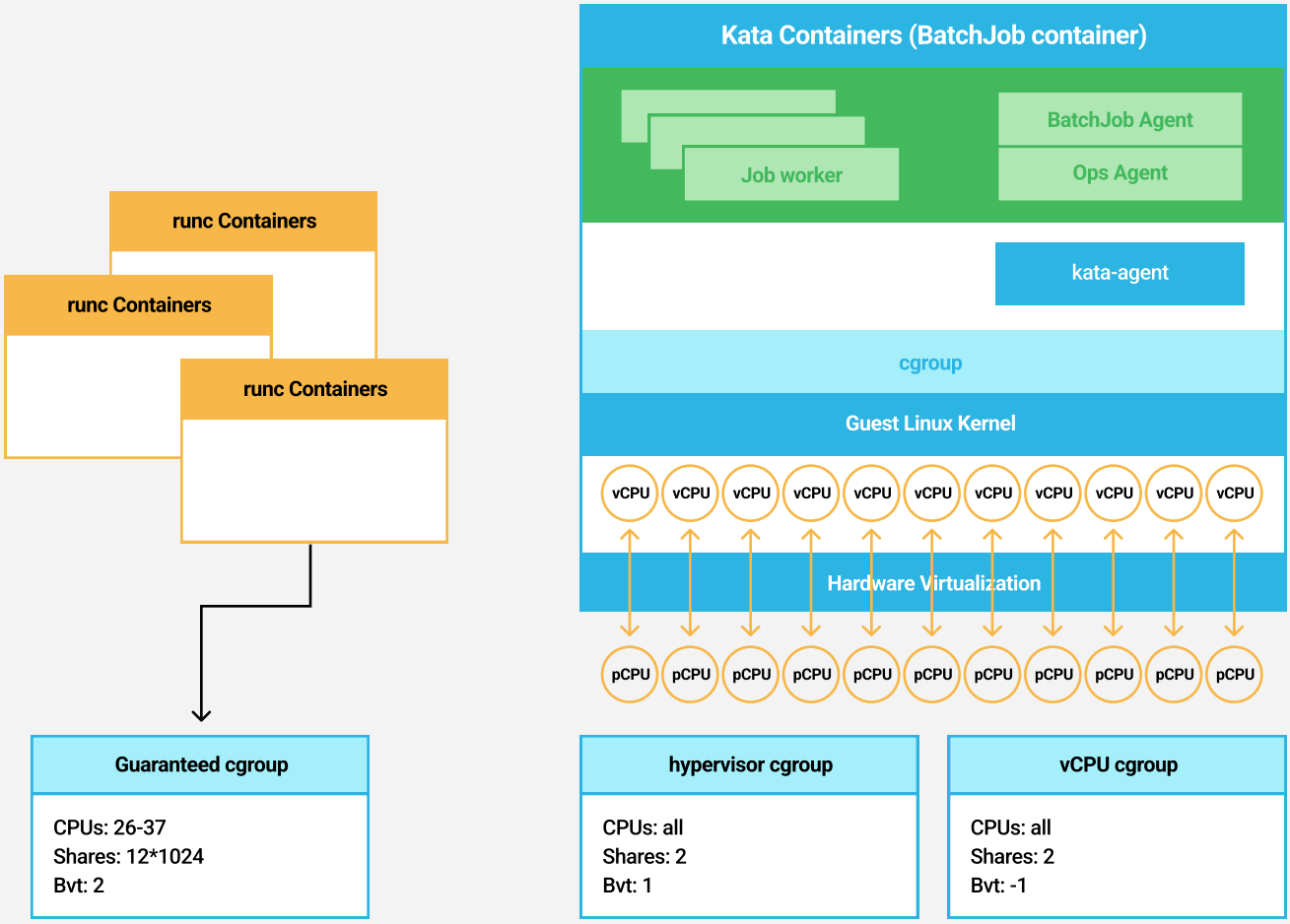

Figure 5: CPU Multi-priority Support

When the Pod of online service uses a runc container, it is bound to different CPU cores through cpuset, and enough shares are allocated to ensure that the CPU resources in the configured cpuset can be fully used.

Offline Pods run in Kata Containers; all offline Pods are in the same CPU cgroup, where cpuset is set to run offline Pods on any CPU, while limiting cpushare to a small value.

The hypervisor's vCPU threads run in a separate CPU cgroup within the Pod CPU cgroup, allowing use of all CPUs. At this time, it needs to use the host-guest vCPU binding technology.

The open-source Kata Containers solution can define CPU shares but cannot meet the special requirements of BVT, RDT, and network packet marking in Ant Group co-location systems. Also on the host, Kata Containers runtime needs group management of multiple resources and group binding of Qemu threads (corresponding to different vCPUs). In this way, different guest vCPUs can correspond to different priority capabilities. At the same time, Kata Containers will tell related information to the batch job scheduling framework in the guest through agent to support different resource priorities.

CPU Isolation Enhancement

Relying on the standard CPU isolation technology provided by the above-mentioned Linux kernel can only ideally preempt resources from offline jobs and obtain more running time, thereby ensuring the stability of online services. However, relying solely on these standard kernel functions cannot achieve perfect isolation, cannot achieve 100% suppression of offline applications by online applications, and cannot completely eliminate the jitter problem of online services. Kata Containers and the kernel team have cooperated to develop some new kernel features, or make targeted improvements to existing kernel features, to achieve more thorough CPU resource isolation and ensure the stability of co-location.

Preemptive Scheduling

On a logical CPU core, online jobs will periodically sleep, wake up, and execute. When online and offline jobs are co-located and run together, the kernel's CFS scheduler will queue the jobs. Since CFS tends to be absolutely fair, we can imagine that in the worst case, the execution chance of online jobs may be divided into half, which means that the scheduling time of jobs running on this core will be doubled, and the RT will increase in service performance, especially the long tail delay will increase significantly.

The cgroup solution is implemented by using the share and period/quota functions provided by cgroup. Share is to control the different cgroup CPU usage time by configuring weights for different cgroups. Period/quota is able to control the time that a cgroup occupies the CPU per unit time. However, whether it is share or period/quota, there is no way to solve one problem, that is, online applications cannot preempt the CPU of offline applications in time when needed. Therefore, in latency-sensitive service scenarios, only using the cgroup solution will affect the response time of online services.

To solve this problem, the Alibaba kernel team developed a kernel feature called Task Preempt. The task preempt function is implemented based on the BVT (Borrowed Virtual Time) framework. Its core idea is to implement multi-priority support for online and offline applications by processing the vruntime of the current task and the ready task differently during scheduling. This feature is based on the CFS scheduler, which marks the online and offline task groups with bvt respectively. This grouping value can be controlled through the cgroup interface. Preemptive scheduling ensures that online services can more easily preempt the CPU resources of offline jobs when needed.

LLC Isolation

Not all resources can be isolated through cgroups and namespaces, such as CPU cache and memory bandwidth.

Among all CPU caches, the one we value the most and the only one that can be directly isolated is the Last Level Cache (LLC). LLC is shared by all cores under each socket, including HT(Hyper Threading). In a co-location environment, if there is no LLC isolation, offline applications are constantly reading and writing data, leading to a large number of LLC occupancy, resulting in continuous pollution of the online application LLC, affecting data access and even increased hardware interrupt latency and performance degradation.

LLC can be enabled through Intel® RDT (Resource Director Technology) technology.

The Intel® RDT framework includes a series of resource allocation technologies: cache allocation technology (CAT), code and data prioritization (CDP), memory bandwidth allocation (MBA) and memory bandwidth monitoring (MBM), etc. CAT turns the LLC into a resource that supports QoS.

In the kernel, we implement LLC isolation in the form of cgroups based on hardware APIs. In order to ensure online services, we allocate more LLCs for online applications and less LLCs for offline applications, so as to reduce the interference between offline applications and online applications.

Memory Isolation

Memory resources are difficult to share. Different from the time-division multiplexing of CPU resources, even if there is a technology such as Linux page cache, it is difficult to achieve real sharing of memory between different processes, and in most cases, memory also involves I/O read and write operations, including reading page cache and reclaiming dirty page cache.

The contention of the process for memory is different from that of the CPU. As long as the process can apply for enough memory, the memory will be used exclusively by the process during its life cycle. Therefore, the target of memory isolation between online services and offline services is mainly considered from the perspective of memory reclaim strategy and OOM priority.

Online service will have a higher memory priority, while offline service will have a lower priority. That is, when the free memory of the whole machine is relatively small, the system will preferentially reclaim the page cache of the offline jobs first; when the memory of the whole machine is tight and the cgroup OOM is caused, the system will preferentially kill the offline jobs first. In this way, the stable operation of online jobs can be guaranteed under two extreme memory conditions.

The kernel provides dozens of levels of memcg(Memory Cgroups) priorities to support services of different priorities and meet different levels of memory QoS requirements.

Storage Isolation

The I/O path of Linux is very long, involving many other software layers like overlay file system, local file system (such as ext4), page cache, BIO and I/O scheduler. The current kernel cannot perform comprehensive and perfect priority control for I/O requests in diverse and dynamically changing service scenarios, which will cause various I/O interference problems.

The resource contention caused by I/O competition is the biggest factor leading to RT glitches in online applications. Batch computing is a heavy I/O service, which has high I/O requirements and has the greatest impact on the host system. Therefore, the isolation of I/O is the top priority of batch computing service isolation, and is a key part of ensuring smooth online service.

The co-location solution of Kata Containers makes full use of the advantages of the cloud data center and efficiently solves this problem. Ant Group's batch computing co-location environment is completely built on cloud servers running X-Dragons. Each server comes with an NVMe disk and multipleESSDs (Elastic Solid State Disk) to solve the isolation problem of the following two I/O competition hotspots.

- Rootfs isolation: Allocate an independent ESSD to the batch computing container, and create a device mapper device for a dedicated root file system directory. The ESSD based device mapper device is used by Kata Containers through the virtio-scsi device.

- Data disk isolation: The NVMe disk on the X-Dragon server can be allocated exclusively for batch computing and used as data disks. It is used by Kata Containers via hardware passthrough to achieve maximum performance.

In this way, whether it is a data disk or an rootfs disk, the Kata Containers solution solves the mutual interference of online and offline I/O bandwidth by physically isolating the online and offline I/O paths from the host.

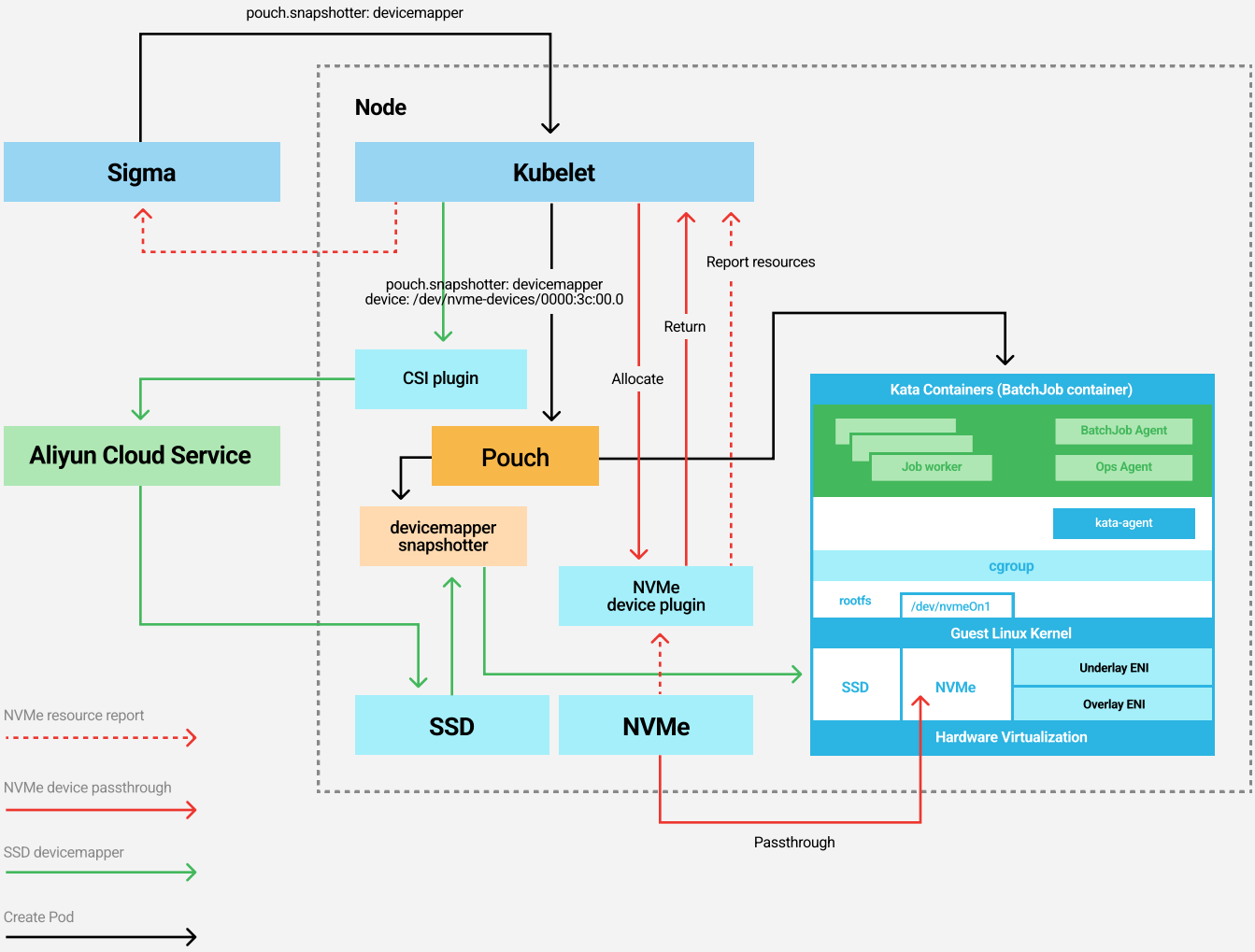

Figure 6: Kata Containers Storage Architecture Diagram

The NVME pass-through design uses Kubernetes' Extended Resource and Device plugin mechanisms and frameworks.

- Extended Resource: Developers can customize resources and report the name and quantity of resources to the API server. The scheduler manages the number of available and allocated resources, updates the resource information when creating and deleting Pods, and selects those nodes with enough available resources when scheduling.

- Device Plugin: Provides a set of common device plug-in mechanisms and APIs, through which device providers can extend hardware devices without modifying the Kubelet code. For example, GPU device support is also implemented with this plug-in mechanism.

The device plugin mechanism allows various manufacturers to report their own devices as Extended Resources to the APIserver of Kubernetes through the mechanism of common plug-ins.

The NVMe device plugin implements the general interface defined by the device plugin. It can use a specific resource name to register hardware resource information in the cluster and is also responsible for the allocation of corresponding NVMe devices.

We deploy the NVMe device plugin as DaemonSet on the servers in the Kubernetes cluster.

Network Isolation

Network Model for Kata Containers

In Kubernetes, CNI is responsible for the network device configuration of the Pod, and runc containers will run in their own network namespace independent of the host one. The traditional method is through virtual ethernet (veth) devices. One end of veth device is placed in the network namespace of the container, and the other end is placed in the network namespace of the host (usually a bridge device). However, this way of relying on namespace is not suitable for hypervisor-based container technology such as Kata Containers, so the way Kata Containers adopts it is to add a TAP device to the virtual machine to enable network interconnection.

To be compatible with how traditional container runtimes are set up for networking, Kata Containers uses TC (Traffic Control) filters to forward traffic between veth devices and TAP devices. As shown below.

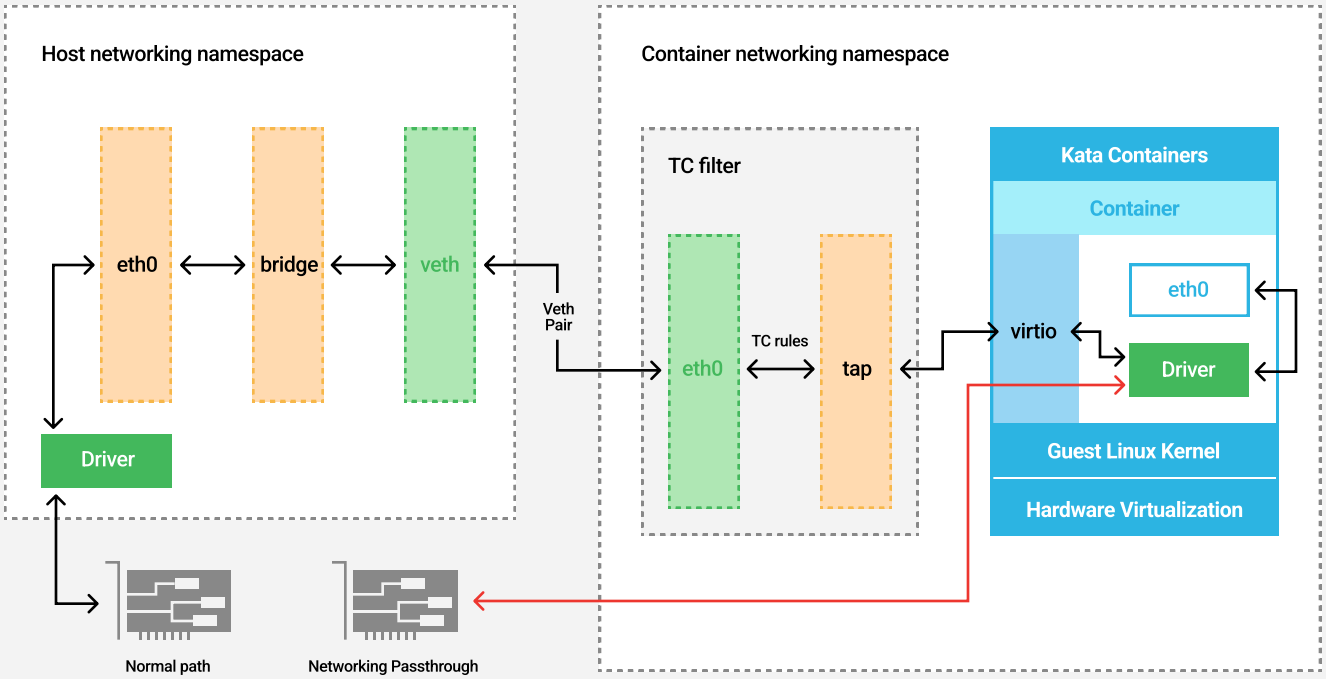

Figure 7: Kata Containers Native Network Model

Kata Containers will use TC filters to forward traffic between the host's container network and the virtual machine, providing network functions for the Pods in the virtual machine. Here CNI will create a veth device named eth0 in the container network namespace; Kata Containers will create a tap device tap0_kata in the virtual machine and set a bidirectional forwarding rule between eth0 and tap0_kata. You can see the entire network forwarding process involving multiple network devices, and its performance loss cannot be ignored.

Batch Computing Co-location Network Solution

Kata Containers co-location clusters run on the cloud and use many cloud infrastructure components. In terms of network, the elastic network card ENI (Elastic Network Interface) device is used. ENI is a virtual NIC provided by Alibaba Cloud that can be bound to VPC-type ECS (Elastic Cloud Server) instances in the private network. Through the ENI card, users can achieve high-availability cluster construction, low-cost failover, and refined network management.

In the Kata Containers solution, thanks to the various convenient functions provided by the cloud infrastructure, batch computing containers are mainly used to achieve network connectivity and isolation through the following technologies, providing high-performance network communication services:

- Two ENI network cards are applied on X-Dragon, both of which are directly connected to the guest.
- One ENI is used for the overlay network, and the batch computing container uses this network card to communicate with other batch computing nodes (Kata Containers).
- The other ENI is used for the underlay network, and batch computing containers use this network card to access the storage cluster.

Through the independent network card provided by X-Dragon, the network card bandwidth interference and performance impact of online applications and offline applications can be avoided.

The implementation of the underlay ENI also adopts the Extended Resource mechanism of Kubernetes, which is implemented through a custom controller.

Important interrupts for pass-through NICs are also bound to high-priority vCPUs by the Kata agent for maximum performance.

Secure Memory Resiliency

The elastic scaling of capacity requires two capabilities, VPA (Vertical Pod Autoscaling) and HPA (Horizontal Pod Autoscaling). The core idea of VPA is to dynamically adjust the resource specifications of the container according to the actual resource usage of the container during container running; the core idea of HPA is to reduce the resource demand during low service peak periods, and reducing the number of containers or services can save costs. At peak times, we automatically increase the number of containers or servers. Both VPA/HPA provide the lowest-cost infrastructure elastically based on actual traffic changes or resource demands, thereby reducing resource usage costs.

Based on the complementarity of online service and batch computing offline service in terms of time, we have developed a time-sharing scheduling system for Kubernetes, that is, when the online traffic peaks, the amount of batch computing jobs and the resource quota of batch computing containers are reduced, leaving the memory for online services to use. After the online traffic goes down at night, we increase the memory resource request for batch computing and reduce the memory usage of online services, so that more of the memory goes to offline services for batch computing. In this way, resource utilization can be maximized according to different services peak hours.

Kata Containers adopts the virtio-balloon and THP (Transparent Huge Page) solution to provide support for dynamically modifying container memory.

The memory elasticity needs to involve several major components such as Kubernetes, batch computing service, Kata Containers, and time-sharing scheduling system. The following figure is a schematic diagram of the process of adding batch computing Pod memory resources:

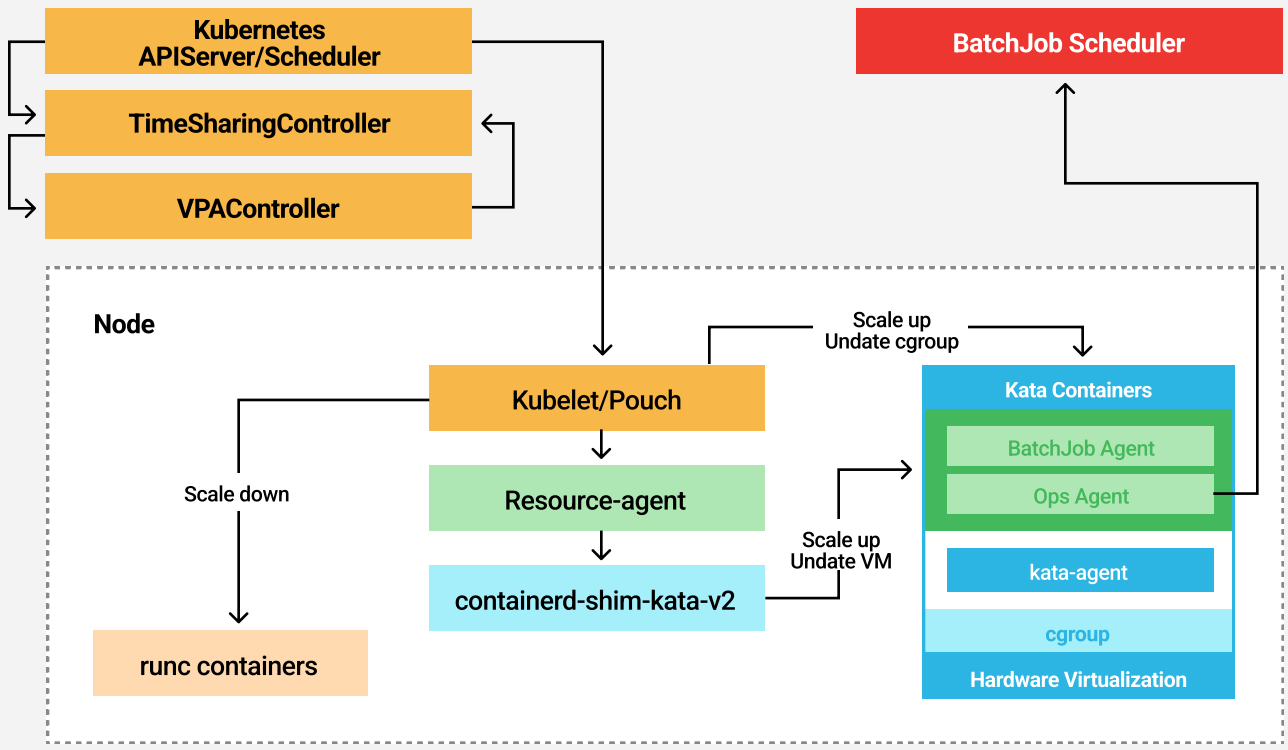

Figure 8: Kata Containers Memory Resiliency Architecture

TimeSharingController is a subsystem of the scheduling system. It will adjust the corresponding container specifications according to the actual water level of resource utilization and the preset rules of the platform, so as to achieve the ability of time-sharing and multiplexing of resources, thereby improving resource utilization of the cluster.

Resource-agent is deployed on each node as a daemonset, expands and shrinks the Kata Containers according to the scaling configuration information issued by the upper layer, and connects to the service control system (Batch Job Agent) in the batch computing container to notify the batch computing dynamically adjust the number of offline jobs on this node.

Operations

A co-location environment must guarantee not only the stability of delay-sensitive online services, but also the stability of offline services with high throughput and high computing performance requirements. A co-location environment must guarantee not only the stability of daily co-location, but also the stability of high traffic during shopping festivals like 618 (June 18, the second-largest e-commerce shopping day in China to double-11) or double-11.

In order to ensure the stability of the daily operation of the co-location system, we should start from two aspects: the first is to take stability as the basis of the system and design stability into all aspects of the system; the second is daily monitoring and alarming, as well as post-event investigation. This part can be strengthened in terms of operations.

Kata Containers Ops and Monitoring Architecture

In order to support more comprehensive and refined indicator collection for Kata Containers, we adopted the monitoring and collection service (kata-monitor component) introduced in the Kata Containers 2.0 release, and collected the metric data of all Kata Containers components from the Kata Containers shim process as the entry point. Its general structure is as follows:

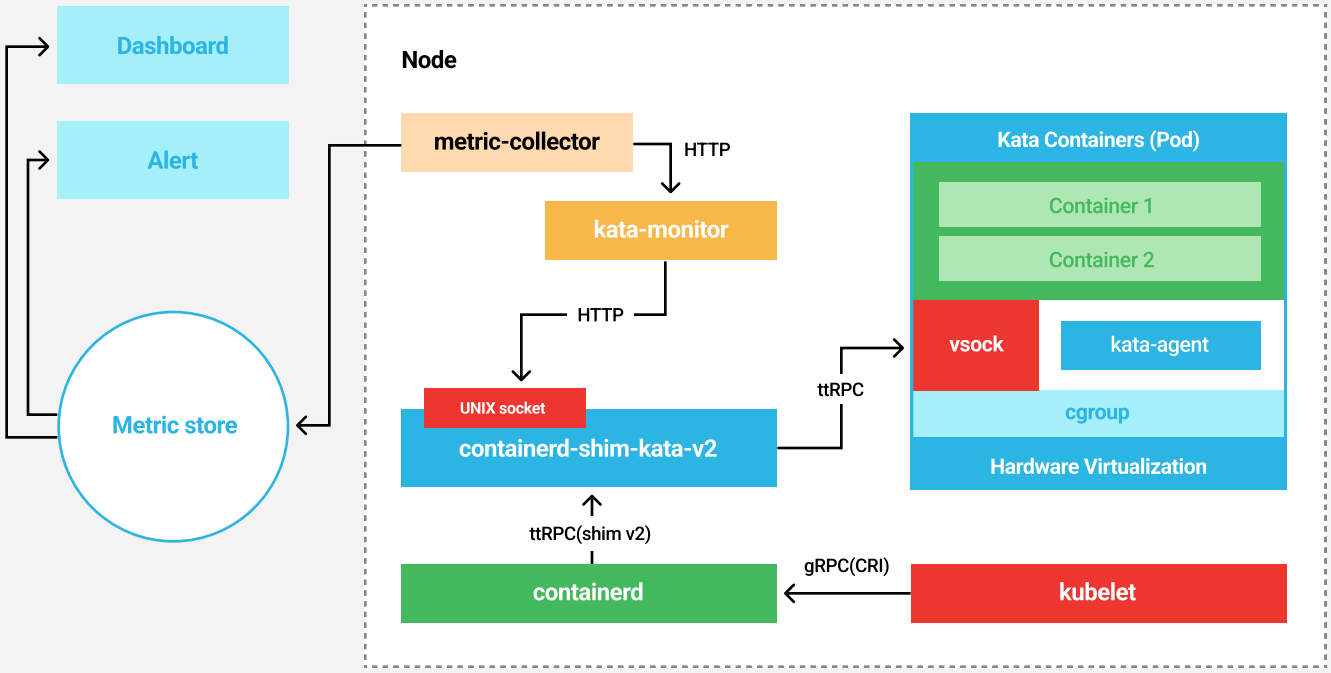

Figure 9: Kata Containers Ops Architecture

The metric collection process running on each node will access the performance metrics data of the Kata Containers through the UNIX socket provided by the Kata Containers shim process. These metrics include:

- Performance data for the shim process itself
- Performance data of the agent process within the guest
- Guest OS performance data
- Hypervisor performance data
- Performance data for containers inside Kata Containers (i.e., performance data for containers inside Pods)

In addition, since Kata Containers uses the Pod overhead feature, this part is also inconsistent with the traditional runc container, and needs to be adapted on the monitoring system to provide users with a unified user experience, so that users do not have to worry about the underlying container runtime.

In addition, in order to support extended operation capabilities, we also provide additional operation interfaces for Kata Containers, which are also provided through Unix sockets. SRE can call these interfaces through command line or programming to complete the custom operations of Kata Containers.

Monitoring

Monitoring is a vital part of the secure container ecosystem. A complete monitoring infrastructure is the foundation for customers to use Kata Containers with peace of mind. It is a necessary condition for Kata Containers to be deployed and applied on a large scale within the company. Kata Containers has also been integrated with the company's monitor ecosystem and to provide complete and accurate monitoring services.

We improve the stability analysis and monitoring system construction in the co-location scenario from the following aspects.

Physical server level

The existing online service operation platform can be used to analyze and monitor the resource overhead of the host machine, to guarantee the stability of physical server operations.

Container level

Through the container-oriented infrastructure monitoring platform, we actively monitor the system load monitoring of application containers. And the monitoring and collection instructions are adjusted to more accurately collect the real resource usage of containers in the co-location environment.

Linux Kernel Extensions

The kernel supports resource isolation and priority guarantee, but in the case of co-location, we also need to display the usage of various resources in a more accurate form to evaluate whether resource isolation and guarantee can meet the expectations.

Therefore, on the basis of the original Linux kernel, we have extended the CPU and memory performance data collection by supporting customized real-time kernel data collection in units of cgroups.

CPU Metrics

The core problem of CPU resources is scheduling contention and scheduling delay. We increase the running time (steal time) of the container being interfered by other CPU cgroups shared through CPU share and also supplement the interference data among groups under the same level or other level groups. In addition, in order to refine the scheduling delay time, the container cgroup scheduling delay distribution histogram data is also provided.

Memory Metrics

The main impact of memory on online applications is reflected in direct memory reclaim, so we have enhanced the container cgroup level and host level and memory reclaim-related metrics, including:

- Memory usage exceeded the configured number of times.
- The number of times direct reclaim occurred.
- The histogram of the distribution of processing time for the latency distribution of direct reclaim. (This data can help us locate the delay and the main reasons for the occurrence of direct page reclaim.)
- The number of scanned memory pages scanned by direct reclaim and the number of successfully recycled pages, which can help measure the pressure of memory reclaim.

Operations

Before adopting Kata Containers, the two major scheduling systems (batch jobs scheduling system/Kubernetes) need to coexist on the same physical server and need to solve their respective system/service environment dependencies, system/service performance monitoring issues, and daily operations changes, rapid service deploy, and host maintenance issues, etc. To be consistent in terms of the online and offline basic operation management and control, the same problems are required to be unifiedly handled, while different problems are handled in a differentiated way. Seeking common ground while reserving differences and co-existing, it is necessary to integrate the collaborative operations of multiple teams such as Ops agent, co-location control, online and offline services teams.

After the adoption of Kata Containers, since batch computing services are run in secure containers and completely isolated from the host, the complexity of host operation is reduced, and the operation interference between online and offline services disappears, so as to achieve isolation at the operation level.

After all, Kata Containers provides a container-based user interface, which is different from a traditional host-based operation. In order to meet the operation needs of the batch computing team, Kata Containers is also specially designed for batch computing service, offering extended operation functions so that it will be convenient for the SRE team to maintain, manage and upgrade batch computing containers.

Extensions

In order to support more functions to extend Kata Containers, we have added some additional management interfaces to the kata shim (containerd-shim-kata-v2) process, such as memory elasticity. In terms of operations, Kata Containers has also made some enhancements and improvements to be compatible with existing co-location and internal observability platforms.

- Non-stop service hot upgrade of each key component runtime and agent
- Post-panic vmcore capability
- Controlling process pprof

Through these enhancements, we make sure that Kata Containers run properly under the visual platform, and provide an undifferentiated operation experience for traditional SREs.

Production Effect

Since Kata Containers uses hypervisor-based isolation technology, compared to the shared kernel method such as runc, it will definitely bring a certain performance overhead at the VMM layer. If this part of the overhead is too large, it will also affect the implementation of the Kata Containers co-location solution.

Therefore, we have specified strict practice conditions to ensure that the adoption of Kata Containers will not bring too much performance regression to the system. We have developed some core metrics to measure the performance loss and production effect of Kata Containers, specifically the following three:

Jitter: Ensure the stability of response time for online services (that is, completely eliminate the impact of offline delay on online service and reduce online service jitter).
Utilization (cost): Exceeds the resource utilization of the old co-location.
Performance loss: close to the energy efficiency ratio of the runc container without online suppression.

Through the solidarity and cooperation of the participating teams in the co-location project, we finally achieved all the above goals, and through the co-location of large-scale batch computing, the average daily resource utilization of servers has been improved several times, saving a lot of infrastructure resource costs, and meeting the performance, delay and cost goals set at the beginning of the co-location project.

Stability

Online Kernel Scheduling Long Tail Latency

The co-location based on Kata Containers adopts an independent kernel to strengthen isolation, thus reducing the competition between kernels on the host. From the perspective of kernel scheduling delay, under the same application and pressure scenario, Kata Containers is able to reduce more than 80% of the scheduling delays that are longer than 20ms, which is very meaningful for ensuring the RT stability of online applications.

Online Applications RT

RT is a key indicator of online applications performance, and the premise of co-location is not to have a negative impact on the RT of online applications. Through offline analysis of the trace data of the core transaction application, after the aggregation of all trace data according to the interface dimension, it can be seen from the actual effect that the average RT and P99 RT of the online application remain in a stable state as a whole, and there is no performance degradation.

Memory Isolation Effect

The offline services of the original co-location system runs directly on the host kernel. Because the memory allocation and release operations of offline applications are very frequent and require a lot of memory, the host memory often faces drastic changes, causing great pressure on the kernel. On the other hand, in Kata Containers, the overall memory usage of offline containers is relatively stable, and the memory usage of the host is also very stable, avoiding many problems caused by memory allocation and release.

Downtime Metrics

From the system monitoring point of view, due to its excellent fault isolation effect, the downtime of Kata Containers is much lower than that of the old co-location system, and the stability problem through service feedback is almost reduced to zero.

Decoupled Ops System

In the traditional co-location method, the same physical server is managed by multiple systems: online Kubernetes and offline batch computing control platforms, each using different operation platforms and tools, sharing host resources, and highly coupled, thereby bringing many conflict problems, which increase the complexity of operation, and at the same time, it is more prone to stability issues.

After Kata Containers is adopted, the complexity of the management of the same physical server resources by multiple parties is solved at the operation level, and the risk of operation accidents is reduced. Online and offline services are independently operated without interfering with each other. Kubernetes can manage batch computing containers like online applications, and batch computing services can also manage batch computing secure containers like a physical server, thus avoiding a series of traditional operation problems such as software conflicts.

Carbon Neutrality

Green and low carbon has received extensive attention from technology companies in recent years and has been put into practice. Ant Group has also announced its own carbon neutral roadmap. By adopting three resource scheduling technologies such as "online/offline co-location technology," "cloud native time-sharing scheduling," and "AI elastic capacity," which are independently developed by Ant Group, we are able to achieve the sharing of computing power through those types of "green computing" technologies, reduce the carbon emissions of the data center, and build a green data center in the future.

Ant Group has achieved carbon neutrality for scope 1 and 2 emissions from its operations in 2021. In part of scope 3, we also help the upstream data center in the supply chain achieve a carbon emission reduction of 29591.48 t CO2e through "green computing" technology.

Acknowledgments

Batch computing co-location based on Kata Containers is a huge systematic project, spanning the two major organizations of Ant Group and Alibaba Group, involving Ant Group Security Container Team, Infrastructure Team, Kubernetes Team, and Batch Computing SRE Team, as well as the Alibaba Cloud kernel and operating system teams. It can be said that this is the result of the company's cross-team cooperation and a successful example of cross-team cooperation projects. The success of the co-location project is the outcome of the technical innovation and project management capabilities of the various teams of the group as a whole.

The writing of this white paper is not only supported by the colleagues from the secure container and storage team in Ant Group's Trusted Native Technology Department, but also cannot be separated from the hard work of colleagues who participated in the review and approval of this white paper. Sincere thanks to everyone who helped write this white paper.

Contributors

Written by
Bin Liu, Tao Peng and the Kata Containers team of Ant Group

Reviewed by
Haoyang Li, Allison Price and Jennifer Fowler